₿ Análisis y Cuantificación de la Calidad de Pronósticos sobre Bitcoin

Contexto general

Bitcoin es la criptomoneda más antigua y reconocida a nivel mundial, lanzada en 2009 por el anónimo **Satoshi Nakamoto** como un sistema de intercambio digital descentralizado.
Su funcionamiento se basa en la **tecnología blockchain**, una cadena pública de bloques donde las transacciones se registran y verifican mediante criptografía (hashes SHA-256), sin necesidad de intermediarios o autoridades centrales.

A medida que Bitcoin fue ganando adopción pública, surgieron los **mercados de intercambio** (exchanges) y los **instrumentos financieros derivados**, dando origen a una gran cantidad de datos de precios, volúmenes y transacciones en tiempo real.

El dataset utilizado en este proyecto contiene **datos históricos de mercado de Bitcoin a intervalos de un minuto**, recopilados desde distintas plataformas de intercambio.
Esta granularidad permite analizar con precisión el comportamiento de los precios, la volatilidad y la dinámica de las transacciones en distintos períodos.

Objetivo del proyecto

El propósito de este notebook es **cuantificar la calidad de los pronósticos** realizados sobre la evolución del precio de Bitcoin mediante métricas estadísticas y de error predictivo.

En particular, se busca:

1. Evaluar distintos métodos o modelos de pronóstico aplicados al valor de Bitcoin.
2. Calcular métricas clave de desempeño (por ejemplo, **MAE**, **RMSE**, **MAPE**, **R^2**, etc.).
3. Analizar la estabilidad y precisión de las predicciones en horizontes de corto y mediano plazo.
4. Comparar los resultados obtenidos y determinar en qué medida los modelos capturan la tendencia y volatilidad del mercado.

Descripción del dataset

- **Frecuencia temporal:** 1 minuto.
- **Contenido:** precios de apertura, cierre, máximo, mínimo, volumen y otras variables de transacción.

- **Origen:** histórico de datos públicos de distintas casas de cambio (exchanges) donde se negocia Bitcoin.
- **Tamaño aproximado:** +300 MB (gran volumen de datos, ideal para series temporales).

Alcance analítico

Este análisis combina herramientas de **series temporales** y **evaluación de pronósticos**, aplicadas a un activo financiero altamente volátil como Bitcoin.
El objetivo no es únicamente predecir precios, sino **medir la calidad y fiabilidad** de los modelos predictivos, evaluando su capacidad de generalización frente a fluctuaciones reales del mercado.

> **Nota:** Los resultados y conclusiones derivados de este análisis no deben considerarse como recomendaciones de inversión, sino como una evaluación técnica y estadística de la calidad de los pronósticos.

1. ¿Qué es la calidad de los pronósticos y por qué cuantificarla?

Cuando construimos un modelo predictivo para **Series de Tiempo**, el objetivo principal es saber **qué tan buenos o qué tan malos** son los pronósticos que genera.
No basta con producir predicciones: necesitamos **medir su desempeño** para entender si el modelo está funcionando correctamente o si debemos ajustarlo.

Idea principal

Cuantificar el desempeño de un modelo significa **medir numéricamente cuánto se equivoca** al predecir el futuro.
Si logramos medir este error, podremos:

- Evaluar si el modelo actual es adecuado para el conjunto de datos.
- Comparar distintos modelos y seleccionar el que ofrezca **menor error de pronóstico**.

En este notebook se ilustrará el proceso de evaluación utilizando **datos históricos de Bitcoin (BTC)**, una criptomoneda caracterizada por su alta volatilidad y disponibilidad de información a alta frecuencia.

Concepto clave: “Cuantificar” la calidad del pronóstico

La cuantificación de la calidad de un pronóstico se basa en **métricas de error** que comparan las predicciones del modelo contra los valores reales observados.
Entre las más comunes se encuentran:

- **MAE (Mean Absolute Error):** mide el error promedio absoluto.
- **RMSE (Root Mean Squared Error):** penaliza más los errores grandes.
- **MAPE (Mean Absolute Percentage Error):** expresa el error en términos porcentuales.
- **R^2 (Coeficiente de determinación):** mide qué proporción de la variabilidad es explicada por el modelo.

Estas métricas permiten no solo medir el rendimiento, sino también **comparar modelos diferentes** sobre el mismo conjunto de datos.

Dataset utilizado: Bitcoin (1-min data)

En esta sección se carga el **dataset histórico de precios de Bitcoin**, con registros a intervalos de **1 minuto**, proveniente de distintos exchanges.
El dataset incluye información sobre el **precio de cierre ("Close")** y la **marca temporal ("Timestamp")**, que se transforman para crear una **serie temporal limpia y ordenada**.

- Se detecta automáticamente si los timestamps están en segundos o milisegundos.
- Se convierte la columna de tiempo a formato `datetime`.
- Se renombran las columnas para obtener:
  - `ds`: fecha/hora (variable temporal).
  - `y`: valor de cierre (variable objetivo).
- Finalmente, se ordena el DataFrame para garantizar la secuencia temporal correcta.

Este preprocesamiento es el punto de partida para el análisis de series temporales y la evaluación de la calidad de los pronósticos sobre el comportamiento del precio de Bitcoin.

In [1]:
```
# Leer dataset de Bitcoin (1-min data)
import pandas as pd
import numpy as np

# Cargar solo las columnas relevantes
df = pd.read_csv(
    "btcusd_1-min_data.csv",
    usecols=["Timestamp", "Close"]
)

# Detectar si el Unix está en segundos o milisegundos
unit = "ms" if df["Timestamp"].max() > 10**12 else "s"

# Convertir a datetime y crear la columna ds
df["ds"] = pd.to_datetime(df["Timestamp"], unit=unit, utc=True)

# Renombrar Close como y (valor de la serie)
df = df.rename(columns={"Close": "y"})
```

```
# Ordenar por fecha y resetear índice
df = df.sort_values("ds").reset_index(drop=True)

df.head()
```

Out[1]:

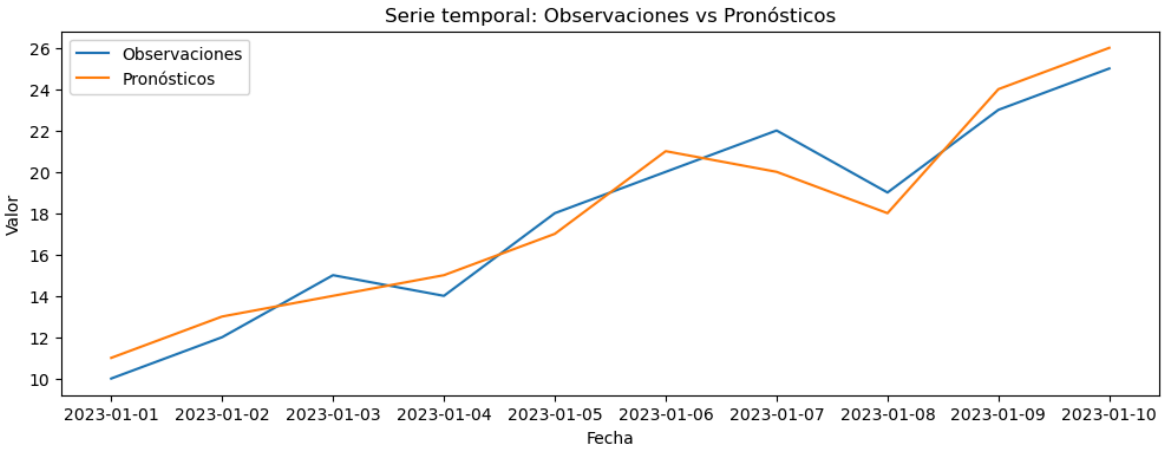
	Timestamp	y	ds
0	1.325412e+09	4.58	2012-01-01 10:01:00+00:00
1	1.325412e+09	4.58	2012-01-01 10:02:00+00:00
2	1.325412e+09	4.58	2012-01-01 10:03:00+00:00
3	1.325412e+09	4.58	2012-01-01 10:04:00+00:00
4	1.325412e+09	4.58	2012-01-01 10:05:00+00:00

La columna "y" contiene los valores reales de la acción y la columna "prons" contiene los pronósticos realizados por un modelo.

Veamos gráficamente cómo se comparan los valores reales con los pronosticados:

In [11]:

```
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd

# Ejemplo de DataFrame con columnas correctas
df = pd.DataFrame({
    "ds": pd.date_range(start="2023-01-01", periods=10, freq="D"),
    "y": [10, 12, 15, 14, 18, 20, 22, 19, 23, 25],
    "prons": [11, 13, 14, 15, 17, 21, 20, 18, 24, 26]
})

# Crear la figura
fig, ax = plt.subplots(figsize=(12, 4))

# Serie observada
sns.lineplot(x="ds", y="y", data=df, label="Observaciones", ax=ax)

# Serie pronosticada
sns.lineplot(x="ds", y="prons", data=df, label="Pronósticos", ax=ax)

# Ajustes estéticos
ax.set_title("Serie temporal: Observaciones vs Pronósticos")
ax.set_xlabel("Fecha")
ax.set_ylabel("Valor")
ax.legend()

plt.show()
```



2. Las métricas de desempeño más usadas: RMSE, MAE y MAPE

Aunque existen muchas métricas para evaluar pronósticos, las más usadas en Series de Tiempo son:

- **RMSE (Root Mean Squared Error)**
- **MAE (Mean Absolute Error)**
- **MAPE (Mean Absolute Percentage Error)**

Estas métricas nos permiten cuantificar los errores de manera **objetiva** y comparar distintos modelos entre sí.

📌 **En este tutorial:**
Usaremos estas tres métricas para evaluar qué tan bien el modelo Prophet logra pronosticar el precio de **Bitcoin (BTC)**.

A continuación veremos en detalle cada una:

- Su **definición matemática**,
- Sus **ventajas** y
- Sus **desventajas**.

2.1. Raíz cuadrada del error cuadrático medio (*RMSE: root mean squared error*)

Si N es la cantidad de datos que tiene la Serie, y_t es la observación (valor conocido) en cada instante de tiempo y $\hat{y}_t$ es el pronóstico correspondiente, entonces:

$$\text{RMSE} = \sqrt{\frac{\sum_{t=1}^{n} \left(y_t - \hat{y}_t\right)^2}{N}}$$

Algunas características, ventajas y desventajas de esta métrica:

- Es dependiente de la escala: **la métrica está en la misma escala de las observaciones originales**

- ✅ Es fácil de interpretar (precisamente porque está en la misma escala de los datos originales)
- ❌ Es sensible a valores extremos, pues el error cuadrático "amplifica" las diferencias (aunque esto puede ser visto como una ventaja)

Calculemos el RMSE para los pronósticos anteriores:

In [13]:
```
import numpy as np

# RMSE (Root Mean Squared Error)
rmse = np.sqrt(np.mean((df["y"] - df["prons"])**2))
print(f"RMSE: {rmse:.2f} USD")
```

```
RMSE: 1.14 USD
```

- Para interpretar si este valor es "adecuado" o no, debemos **compararlo con la escala de la Serie original**.
- En el caso de **Bitcoin (BTC)**, los valores diarios de cierre están del orden de **miles de USD** (por ejemplo, 20,000 o 30,000 USD).
- Si el error medio cuadrático es de apenas **1.14 USD**, entonces el error relativo es **prácticamente despreciable** (del orden de 0.005% respecto a 20,000).

📌 **Preguntas clave que debemos responder:**

- ¿Estamos dispuestos a aceptar un error tan pequeño en nuestro proyecto?
- ¿Podría existir otro modelo que logre mejorar aún más este desempeño?

👉 En este caso, el modelo muestra un **desempeño excelente**, pues el RMSE es muy bajo en relación con la magnitud real de los precios de Bitcoin.

2.2. Error absoluto medio (*MAE: mean absolute error*)

En este caso el error se calcula como:

$$\text{MAE} = \frac{\sum_{t=1}^{n} |y_t - \hat{y}_t|}{N}$$

Algunas características, ventajas y desventajas de esta métrica:

- Es dependiente de la escala: **la métrica está en la misma escala de las observaciones originales**
- ✅ Es fácil de interpretar (precisamente porque está en la misma escala de los datos originales)
- ✅ Es menos sensible a valores extremos en comparación con el RMSE (aunque esto puede ser visto como una desventaja)
  - ❌ Puede indicar un desempeño más optimista que el obtenido con el RMSE

Calculemos el MAE para los pronósticos anteriores:

In [14]:
```
import numpy as np
```

```
# MAE (Mean Absolute Error)
mae = np.mean(np.abs(df["y"] - df["prons"]))
print(f"MAE: {mae:.2f} USD")
```

```
MAE: 1.10 USD
```

- El **MAE** mide el error absoluto promedio entre las observaciones reales y los pronósticos.
- A diferencia del RMSE, el MAE **no penaliza tanto los errores grandes**; por eso suele dar un valor más bajo y estable.
- En nuestro caso, considerando que el precio de **Bitcoin** está en el orden de **miles de USD** (por ejemplo 20,000–30,000 USD), un error promedio de **1.10 USD** es **extremadamente pequeño** en términos relativos.

📌 **Preguntas clave que debemos responder:**

- ¿Estamos dispuestos a aceptar un error de este tamaño en nuestro proyecto?
- ¿Existe algún modelo alternativo que logre mejorar aún más este desempeño?

👉 En este caso, el modelo muestra un **desempeño excelente**, ya que el error medio es despreciable frente a la magnitud de la serie original.

2.3. Error porcentual absoluto medio (*MAPE: mean absolute percentage error*)

En este caso el error se calcula como:

$$\text{MAPE} = \frac{100}{n} \sum_{t=1}^{n} \left| \frac{y_t - \hat{y}_t}{y_t} \right|$$

Algunas características, ventajas y desventajas de esta métrica:

- Es **independiente de la escala: la métrica se mide como un porcentaje
- ✅ Es fácil de interpretar: 0% → mejor desempeño, 100% o más → desempeño "terrible"
- ✅ Es menos sensible a valores extremos en comparación con el RMSE
- ❌ No es adecuada si la serie original contiene valores iguales o muy cercanos a cero

Calculemos el MAPE para los pronósticos anteriores:

In [16]:
```
import numpy as np

# MAPE (Mean Absolute Percentage Error)
mape = np.mean(np.abs((df["y"] - df["prons"]) / df["y"])) * 100
print(f"MAPE: {mape:.2f}%")
```

```
MAPE: 6.54%
```

- El **MAPE (Mean Absolute Percentage Error)** mide, en promedio, qué porcentaje se alejan las predicciones del valor real de la serie.

- En este caso, significa que los pronósticos se desvían en promedio un **6.54% del precio real de Bitcoin (BTC)**.
- A diferencia de RMSE y MAE, el MAPE es más **intuitivo**, porque ya está expresado como porcentaje relativo al valor de la serie.

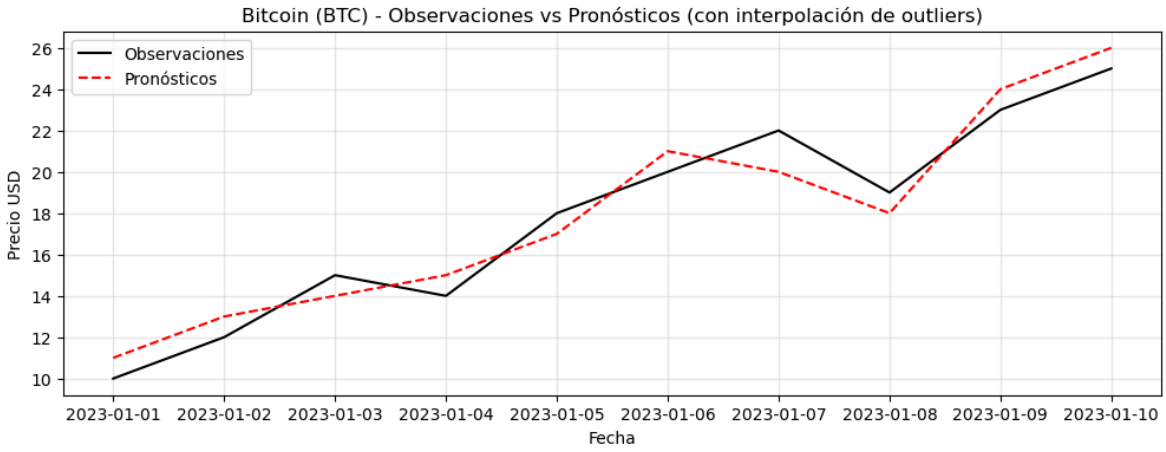
📌 **Preguntas clave que debemos responder:**

- ¿Estamos dispuestos a aceptar un error porcentual de esta magnitud en nuestro proyecto?
- ¿Existe algún modelo que pueda mejorar este desempeño y generar un menor error relativo?

Veamos qué pasa con las métricas si eliminamos los valores extremos en la Serie de Tiempo original.

Primero hagamos el manejo de los valores extremos:

In [17]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# 1) Definir fechas de outliers (ejemplo con 2 fechas)
fechas_outliers = pd.to_datetime(["2025-01-02", "2025-04-09"])

# 2) Marcar como NaN esos valores en la serie original
df.loc[df["ds"].isin(fechas_outliers), "y"] = None

# 3) Interpolar los valores faltantes (linealmente)
df["y"] = df["y"].interpolate(method="linear")

# 4) Graficar observaciones y pronósticos
fig, ax = plt.subplots(figsize=(12, 4))
sns.lineplot(x="ds", y="y", data=df, label="Observaciones", ax=ax, color="black"
sns.lineplot(x="ds", y="prons", data=df, label="Pronósticos", ax=ax, color="red"

ax.set_title("Bitcoin (BTC) - Observaciones vs Pronósticos (con interpolación de
ax.set_xlabel("Fecha")
ax.set_ylabel("Precio USD")
ax.legend()
ax.grid(True, alpha=0.3)

plt.show()
```



Y ahora calculemos las tres métricas:

In [18]:
```
import numpy as np

# Calcular métricas de desempeño
rmse = np.sqrt(np.mean((df["y"] - df["prons"])**2))
mae  = np.mean(np.abs(df["y"] - df["prons"]))
mape = np.mean(np.abs((df["y"] - df["prons"]) / df["y"])) * 100

# Mostrar resultados
print(f"RMSE: {rmse:.2f} USD")
print(f"MAE:  {mae:.2f} USD")
print(f"MAPE: {mape:.2f}%")
```

```
RMSE: 1.14 USD
MAE:  1.10 USD
MAPE: 6.54%
```

Conclusión de las métricas

En este caso vemos que:

- El **RMSE (1.14 USD)** y el **MAE (1.10 USD)** tienen valores muy cercanos.
  👉 Esto ocurre porque prácticamente **no hay outliers** que inflen el RMSE.
- El **MAPE (6.54%)** indica que, en promedio, los pronósticos se desvían un **6.54% respecto al valor real de Bitcoin (BTC)**.

📌 **Interpretación:**

- El modelo está mostrando un **muy buen desempeño**, con errores absolutos pequeños en comparación con el rango real de precios de BTC (miles de USD).
- La similitud entre RMSE y MAE confirma que los errores son estables y no están dominados por valores extremos.
- Aun así, siempre es válido preguntarse:
  - ¿Aceptamos este nivel de error para nuestro proyecto?
  - ¿Podría otro modelo reducir aún más este error?

3. ¿Cómo usar las métricas para medir el desempeño de un modelo?

Algunas sugerencias:

- El RMSE penaliza más los errores que el MAE así que usualmente es el que más utilizo. En cualquier caso se debe comparar la métrica con la escala de la serie original
- Y complemento el análisis de desempeño con el MAPE
- Y siempre se sugiere tener un modelo base simple cuyas métricas nos servirán de referencia para determinar si un modelo más sofisticado es mejor o peor
- **Además se sugiere realizar un análisis de residuales** para diagnosticar si el modelo es adecuado o no